Counting Sort.

Note Title 11/8/2007

```
//input must be nonnegative ints
// A is input, B is output

//for each input A[i], find
// all values less than it.
(0) Let MaxVal = max(A);
(1) Initialize all C[0 .... MaxVal ] to zero
(2) For j = 1 to n
        C[[A[j]] ++;
    // C[i] is # elements in A equal to i
(4) For i = 1 to MaxVal
        C[i] = C[i] + C[i-1].
    // Now C[i] is # elements in A ≤ i

(6) For k = n to 1
        Let key = A[k].  //value in A
        Let count = C[key]. // # elements ≤ val
        B[count] = key;
        C[key] = count - 1;
```

Counting sort example

input A:

2	1	6	5	2	5	3	1
1	2	3	4	5	6	7	8

(0) max Val = max(A) = 6

(1) Initialize all C[0...6] to Ø

C

Ø	Ø	Ø	Ø	Ø	Ø	Ø
Ø	1	2	3	4	5	6

(2) Let C[i] = # elements in A equal to i

C

Ø	2	2	1	Ø	2	1
Ø	1	2	3	4	5	6

(4) Let C[i] = # elements in A $\leq$ i

C

Ø	2	4	5	5	7	8

6) Let $B[1 \ldots n]$ be the OUTPUT array.
Note: if $C[i]$ has the # elements $\leq i$,

then $B[C[i]] = i$

count of all values $\leq i$

example, if $C[3] = 4$,
then there are 4 values ≤ 3,
so put a 3 in the 4th slot in B.

Work backwards from rightmost input element to leftmost input element.

input A:

2	1	6	5	2	5	3	1
1	2	3	4	5	6	7	8

scratch C:

	~~0~~ ~~1~~ ~~2~~ 3		4		6	
∅	~~2~~	~~4~~	~~5~~	5	~~7~~	8
0	1	2	3	4	5	6

output B:

	1						
1	2	3	4	5	6	7	8

	1			3			
				5			

	1			3		5	
						7	

	1		2	3		5	
			4				

stable!

	1		2	3	5	5	
					6		

	1		2	3	5	5	6
							8th.

1	1		2	3	5	5	6

SORTED!

1	1	2	2	3	5	5	6

Counting Sort

What if we also
have negative integers?
or min $\gg \emptyset$?

- compute Min
- compute Max.

- give C size (Max - Min).
  $C[\emptyset \ldots (Max - Min)]$.

- subtract min from all values
- do Counting Sort as before.

- add min to all elements in B.

alternatively, subtract min from values
used as indices to C.

Runtime of Counting Sort (array reads, array writes)

$\Theta(\max(n, \text{MaxVal}))$

or

$\Theta(\max(n, \underbrace{\text{range}}_{\text{MAX}-\text{MIN}+1}))$

equivalent

$= \Theta(n + \text{range})$

if range is $O(n)$,

then algorithm is $\Theta(n)$.

Radix Sort for nonneg. integers

Sort one digit at a time

329	457	329	329
457	657	839	457
657	329	457	657
839	839	657	839

sort rightmost digit first ⟶ sort leftmost digit last

Digit sorts must be STABLE. counting sort is a good choice.

- MaxVal on digit sort is 9.

so digit sort is $O(\max(n,9)) \rightarrow O(n)$. if $n>9$

- so runtime of radix sort is $O(dn)$

$d = \#$ digits

General Radix Sort Runtime

d = # "digits"
(could be other data)

r = range of each digit

n = # values.

Radix sort runtime is $O(d(n+r))$

(equivalent to $O(d \cdot \max(n, r))$)

if d is fixed and $r \in O(n)$,
then radix sort is linear.

Question 1: If we have n b-bit integers, can we sort them in $\Theta(b \cdot n)$ time?

Use radix sort.

\# digits $= b$

range $= 2$ $(\emptyset \text{ or } 1)$

$n = n$

so runtime is $\Theta(\#\text{digits}(n + \text{range}))$

$= \Theta(b(n+2))$

$= \Theta(bn)$

Question 2: How many bits are used to represent the numbers in the range $0 \ldots n-1$?

n	$n-1$	# bits needed	
4	3	2	01 10 11
8	7	3	001 ⋮ 111
16	15	4	0001 ⋮ 1111

$\log_2 n$ bits!

So if sorting n b-bit integers takes $\Theta(bn)$ time, then sorting n $(\log_2 n)$-bit integers takes $\Theta(n \log n)$ time!

Question 3: What if we group the digits into clusters of size r?

$$\begin{array}{ccccc} 23\,\boxed{17} & & \boxed{23}\,17 & & 0896 \\ 65\,\boxed{42} & & \boxed{43}\,21 & & 1239 \\ 43\,\boxed{21} & \rightarrow & \boxed{12}\,39 & \rightarrow & 2317 \\ 08\,\boxed{96} & & \boxed{65}\,42 & & 4321 \\ 12\,\boxed{39} & & \boxed{08}\,96 & & 6542 \end{array}$$

$r = 2$

\# of decimal digits

$\#\text{"digits"} = 2 = 4/r$

$\text{range} = 100 = 10^2 = 10^r$

so $\text{runtime} = \Theta(2(n + 100))$

$= \Theta\left(\frac{4}{r}(n + 10^r)\right)$

If n b-bit digits clustered into clusters of size r, runtime is $\Theta(b/r\,(n + 2^r))$

Radix sort one bit at a time is

$$\Theta(b(n+2)) = \Theta(bn)$$

Radix sort r bits at a time is

$$\Theta(b/r\,(n+2^r)) = \Theta(\tfrac{b}{r}n)$$

Save time, but counting sort now needs more space...
(MaxVal is bigger when $r > 1$)

In decimal example, MaxVal grew from 9 to 99

Claim: Given the number of bits to represent n numbers is $O(\log n)$, then if we do all of the bits in a single grouping, RadixSort runs in $\Theta(n)$ time.

$b \in O(\log n)$

let $r = \log_2 n$ (ie., # of bits)

RadixSort is $\Theta\left(\frac{b}{r}(n + 2^r)\right)$ $\quad \frac{b}{r} = 1$

$\Theta(n + 2^r)$

$\Theta(n + 2^{\log_2 n})$

$\Theta(n + n)$

$\Theta(n)$

Lots of hidden constants + memory.

Can we sort n values that are in the range $\emptyset \ldots n^2$ in $O(n)$ time?

n values in the range $0 \ldots n^2$

ok, let's try regular radix sort

#digits $= d$, $d \leq n^2$

runtime $= \Theta(d(n+r)) = \Theta(d(n+n^2))$

eek!

Another way:

- represent all values in binary
- split binary number in two

example: 2 8 6 18 10 36

decimal	binary
2	000010
8	001000
6	000110
18	010010
10	001010
36	100100

$n = 6$

$n^2 = 36$

SORT		SORT		
000 010	$\longrightarrow$	001 000	000010	2
001 000		000 010	000110	6
000 110		010 010	001000	8
010 010		001 010	001010	10
001 010		100 100	010010	18
100 100		000 110	100100	36

\# "digits" = 2 (# clusters)

maxVal = 111 (binary) or 7 $\leq 2n$

So runtime

is $\Theta(d(n+r)) = \Theta(2(n+(2n))$

$= \Theta(n)$

Bucket Sort

Linear expected time.

Assume elements equally distributed across the range of values.

Create n buckets.

example: numbers between ϕ (inclusive) and 1 (exclusive) $[\phi, 1)$

bucket 1 : $[\phi \ldots 1/n)$

bucket 2 : $[1/n \ldots 2/n)$

bucket n : $[\frac{n-1}{n} \ldots n/n)$

note: all elements in Bucket 1 < any element in Bucket 2.

example: numbers between ϕ and 1000

bucket 1 : $[\phi \ldots 1000/n)$

⋮

bucket n : $[\frac{(n-1)(1000)}{n} \ldots 1000)$

Bucket Sort $(A[1...n])$ {

$\emptyset$ Determine RANGE of values using MAX and MIN

1 Create n equal-sized buckets based on RANGE

2 for $i = 1$ to n
    insert $A[i]$ into
        corresponding bucket

5 Sort each bucket

6 Concatenate buckets

Runtime: MAX, MIN: $\Theta(n)$
put in buckets: $O(n)$
concatenate buckets: $O(n)$
BUT SORT EACH BUCKET?

How long to sort each bucket?

example: 8 3 2 ϕ

range: $\phi \to 8$

$n = 4$

	Bucket 1	Bucket 2	Bucket 3	Bucket 4
	$[\phi - 2)$	$[2 - 4)$	$[4 - 6)$	$[6 - 8]$
	ϕ	3, 2	—	8
SORT	ϕ	2, 3		8

answer ϕ 2 3 8

note: # elements in each bucket very low!!

b/c of our ASSUMPTION of uniform data values!

b/c of our ASSUMPTION of <u>uniform</u>
data values:

expected # of
values per bucket is very small.

example: if max bucket size is 8,
then # comparisons to sort = $\underline{24}$.
sort time is $\underline{24} \times \underline{n}$ lists = $\underline{24n}$.
Total runtime is $\underline{\Theta(n)}$.